ECE2277A Project Report

Name: Jacob Dreyer

Student Number: 251241714

Submission Date: 2022/12/08

Problem Statement

The task for this project was to create a sequential circuit that would cycle through a set of numbers between 1 and 7 and display the number on a seven-segment display. The circuit must only be composed of primitive logic gates and the default flip-flops provided by Quartus. The solution had to be designed in Quartus and verified using the simulation tool. The circuit then had to be coded onto an FPGA board and displayed to the TAs to verify its functionality.

Sequence	5	4	3	1	1	2	7

Table 1: The sequence of integers to be displayed in a clocked loop.

Design Method

The first step to determining the sequential circuit was to create functions that would change the current value, or state, of the circuit to the next value. To better visualise what needed to happen the values for the possible current states and corresponding next states were placed into a table. Because the numbers must be transmitted as binary values the current state and next state each were represented by 5 columns. One with the decimal number for better understanding and one for each bit in the number for functionality up to fifteen. 4 bits was needed because duplicate numbers must point to different values for the next state. In the given case the first 1 must point to 1 and the second 1 must point to 2. A state cannot go to 2 different states because it would not know which one to go to. So, by having 4 bits a one could be added to the 4th (most significant) bit. In this case it means the second 1 is represented by 9. This creates two different states, each with its own corresponding next state. The most significant bit could then be ignored when displaying on the seven-segment display resulting in displaying the original number. The next step was to create k-maps for each bit in the next state to find the value for that bit based on the values of the current state. Because D flip flops were being used for this design the value going into the D flip flop had to be found. D flip flops only store the input D so all that needs to be provided as an input is the value for the next state. So based on the bits of the current state the value of each bit in the next state is determined so it can be fed into the D flip flop. For example, the value of Q1 should be 1 when Q3 equals 1 and Q0 equals 0. The exact, simplified, function for each bit in the next state could be determined by using the figures below and grouping terms. If this is done the formulas can be determined. The function for Q0 is $Q0(t+1) = (Q2' + Q1 + Q0') Q3'$ and was found from Figure 1. The function for Q1 was found from Figure 2 and is $Q1(t+1) = Q3 + Q0'$. Next is the function for Q2 from Figure 3 which is $Q2(t+1) = Q2\ Q0 + Q1\ Q0'$. Finally, Q3 was found using Figure 4 and is only 1 after a single number so its formula is $Q3(t+1) = Q3'\ Q2'\ Q1'\ Q0$ which can be simplified to $Q3(t+1) = Q3'\ Q2'\ Q1'$ because of the don’t care value. These functions can be implemented into circuits where multiplication operations result in AND gates and adding operations implemented as OR gates.

Current State					Next State				
#	Q_3	Q_2	Q_1	Q_0	#	Q_3	Q_2	Q_1	Q_0
1	0	0	0	1	1(9)	1	0	0	1
2	0	0	1	0	7	0	1	1	1
3	0	0	1	1	1	0	0	0	1
4	0	1	0	0	3	0	0	1	1
5	0	1	0	1	4	0	1	0	0
7	0	1	1	1	5	0	1	0	1
1(9)	1	0	0	1	2	0	0	1	0

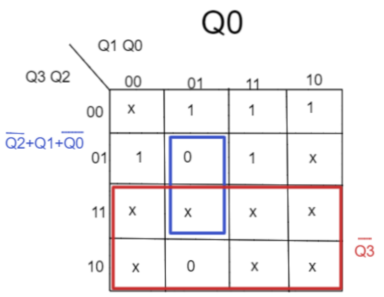


Figure 1: K-Map to determine the value of Q0

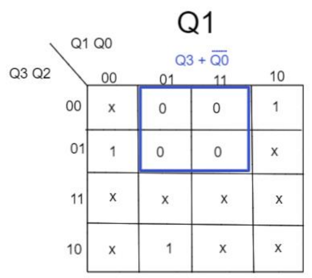


Figure 2: K-Map to determine the value of Q1

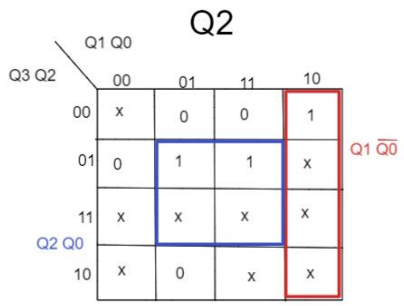


Figure 1: K Map to Determine the Value of Q2

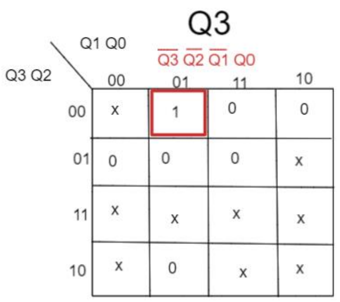


Figure 4: K Map to Determine the Value of Q3

Implementation

The next step in the Project is implementing the design that was already created. The design consisted of 3 major pieces. The first being the flip flops. These store the value that is shown on the display and changes every clock cycle. Next is the Transformation piece. This is the piece that determines the next value for the flip flops. Finally, the last piece converts the binary number to the outputs for the seven-segment display. The functions for the transformation have been determined in the design step and the flip flops are simply 4 different D flip flops. The logic for displaying the number on the seven-segment display is reused from another assignment. Therefore, the pieces only needs to be implemented in Quartus and be linked together. The last step is simply to add inputs and outputs to finish the circuit. The main circuit implementing the 3 pieces is shown in Figure 5. On the left is the input pins for the clock and the reset button. When the reset button is pressed the circuit resets activates the asynchronous set on Q2 and Q0 which sets the output to 5; the first number in the sequence. Every other asynchronous input is set to 1 with VCC so that it operates normally when the switch is not pressed and displays 5 when it is. The clock is the default system clock which is then slowed down by the SlowClock block. This slows down the clock so that it changes the number every second rather than every fraction of a second. The four flip flops are D flips flops that store the values of Q0, Q1, Q2, and Q3 respectively. Their output is then sent to the block which converts the binary number to the 7 outputs required for the seven-segment display. The value of Q3 is not fed in because it is only used to differentiate between two equal values and by ignoring it the actual value is displayed. The QTransformation blocks also take the outputs from the D flip flops and determines what the value of its respective bit should be based on the values of Q0, Q1, Q2, and Q3. It then feeds that value into its associated D flip flop to be displayed the next time it is triggered.

In Figure 6 the circuit that determines the next value of Q0 is shown. In the design stage it was determined that the equation for Q0 was Q0 = (Q2’ + Q1 + Q0’) (Q3’) by using the k-map in Figure 1. This function was then implemented in Quartus and shown in Figure 6.

The Circuit that determines the value of Q1 is shown in Figure 7. Solving the k-map in Figure 2 produced a logic function of Q1 = Q3 + Q0’. Therefore, Q1 is only dependent on the values of Q3 and Q0 which are represented by the two input pins on the left side of the circuit. Q0 is then flipped by going through the not gate to represent Q0’. Also, Q1 only depends on one of these inputs being 1 to output 1. Therefore, only one OR gate is needed in this circuit. The output is then sent to the output pin which represents the next value for Q1.

The value of Q2 is determined by the circuit in Figure 8. Q2 is determined from the inputs Q1, Q2, Q0, and Q0’ which are shown on the right side of the circuit. Q0’ uses the same input as Q0 and is just run through a NOT gate. The next value of Q2 is sent from the output on the right side of the circuit. The circuit was implemented from the function Q2 = (Q1 Q0’) + (Q2 Q0) determined by the k-map in Figure 3.

Next, is the circuit for Q3 shown in Figure 9. Q3 only has a value of 1 in one instance, when the current binary number is 1. This results in a circuit using all the inputs (Q0, Q1, Q2, Q3) to create a function Q3 = Q0 Q1’ Q2’ Q3’. In other words, when Q1, Q2, and Q3 equal zero and Q0 equals one the next value of Q3 is 1 and it is 0 in every other case. In the circuit diagram Q1, Q2, and Q3 are routed through NOT gates before all the inputs are compared using an AND gate.

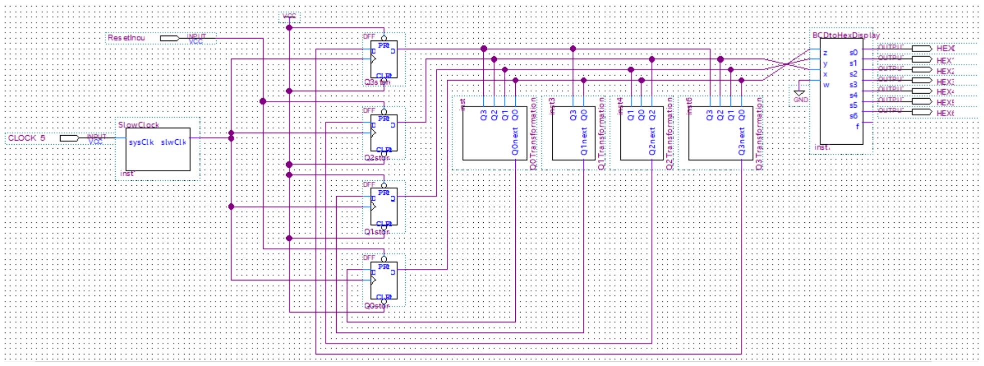


Figure 5: Main Circuit Schematic

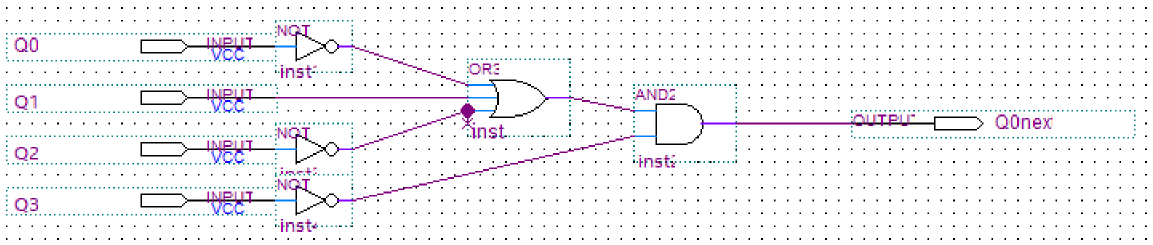


Figure 6: Circuit to Determine the next value of Q0

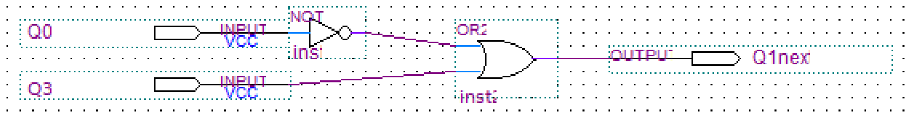


Figure 7: Circuit to determine the next value of Q1

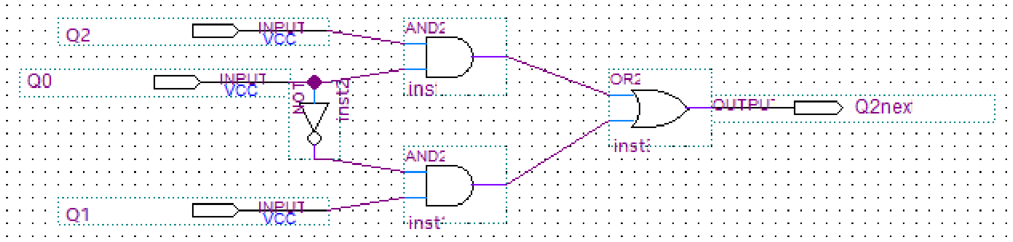


Figure 8: Circuit to Determine the next value of Q2

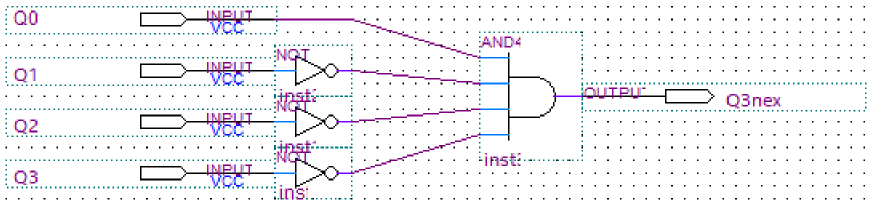
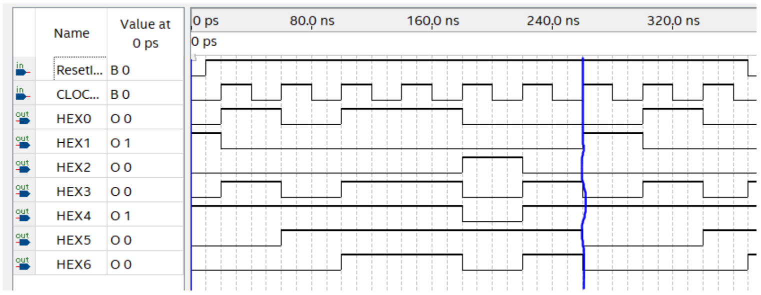


Figure 9: Circuit to Determine the next value of Q3

Lastly, is the waveform shown in Figure 10. This is the simulated output for the circuit based on the programmed inputs. The hex outputs represent each segment on the seven-segment display shown in Figure 11 where Hex0 refers to s0, Hex1 refers to s1, and so on. A particular segment is turned on when it has a value of zero and is turned off when it has a value of 1. So, it's a rising edge element. Going back to the sequence; for each rising edge of the clock the number changes to the next number in the sequence. When the reset input is zero at the start of the sequence the hex display shows the number 5. Once the reset input is set to 1 the circuit starts looping through the sequence showing 4, 3, 1, 1, 2, and 7. This is the point which is shown by the blue line on the waveform and after this the sequence repeats going to 5, 2, and so on.



Name	Value at 0 ps
Resetl...	B 0
CLOC...	B 0
HEX0	O 0
HEX1	O 1
HEX2	O 0
HEX3	O 0
HEX4	O 1
HEX5	O 0
HEX6	O 0

Figure 10: Waveform Simulation of Circuit

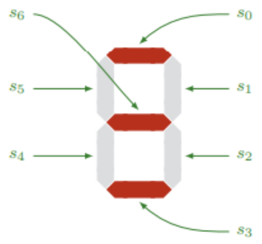


Figure 11: 7 Segment Display with Labels

Discussion

The purpose of this design was to balance out several requirements including complexity, design time, and understanding. The design should result in a simple circuit to implement in Quartus reducing implementation time. By having a simpler Quartus design this also resulted in having a cleaner circuit that is easier to look at and see the flow rather than a design with a lot of wires and being confusing to look at and therefore understand. However, the design should also be simple to design which is why the design used D flip flops. This allows for a relatively quick design stage. This also means that if the sequence is changed it is quicker to redesign. Using D flip flops also meant that only the next state had to be determined, calculated, and fed back into the D flip flops. Overall, the goal for this circuit was to have a design which accomplished the task and was a good balance of a simple Quartus implementation and a simple design process while still allowing for decent flexibility and ease of editing.